

D-Tree Grammars

Owen Rambow

CoGenTex, Inc.
840 Hanshaw Road
Ithaca, NY 14850
owen@cogentex.com

K. Vijay-Shanker

Department of Computer &
Information Science
University of Delaware
Newark, DE 19716
vijay@udel.edu

David Weir

School of Cognitive &
Computing Sciences
University of Sussex
Brighton, BN1 9HQ, UK.
david.weir@cogs.susx.ac.uk

Abstract

DTG are designed to share some of the advantages of TAG while overcoming some of its limitations. DTG involve two composition operations called subserction and sister-adjunction. The most distinctive feature of DTG is that, unlike TAG, there is complete uniformity in the way that the two DTG operations relate lexical items: subserction always corresponds to complementation and sister-adjunction to modification. Furthermore, DTG, unlike TAG, can provide a uniform analysis for *wh*-movement in English and Kashmiri, despite the fact that the *wh* element in Kashmiri appears in sentence-second position, and not sentence-initial position as in English.

1 Introduction

We define a new grammar formalism, called D-Tree Grammars (DTG), which arises from work on Tree-Adjoining Grammars (TAG) (Joshi et al., 1975). A salient feature of TAG is the extended domain of locality it provides. Each elementary structure can be associated with a lexical item (as in Lexicalized TAG (LTAG) (Joshi & Schabes, 1991)). Properties related to the lexical item (such as subcategorization, agreement, certain types of word order variation) can be expressed within the elementary structure (Kroch, 1987; Frank, 1992). In addition, TAG remain tractable, yet their generative capacity is sufficient to account for certain syntactic phenomena that, it has been argued, lie beyond Context-Free Grammars (CFG) (Shieber, 1985). TAG, however, has two limitations which provide the motivation for this work. The first problem (discussed in Section 1.1) is that the TAG operations of substitution and adjunction do not map cleanly onto the relations of complementation and modification. A second problem (discussed in Section 1.2) has to do with the inability of TAG to provide analyses for certain syntactic phenomena. In developing DTG we have tried

to overcome these problems while remaining faithful to what we see as the key advantages of TAG (in particular, its enlarged domain of locality). In Section 1.3 we introduce some of the key features of DTG and explain how they are intended to address the problems that we have identified with TAG.

1.1 Derivations and Dependencies

In LTAG, the operations of substitution and adjunction relate two lexical items. It is therefore natural to interpret these operations as establishing a direct linguistic relation between the two lexical items, namely a relation of complementation (predicate-argument relation) or of modification. In purely CFG-based approaches, these relations are only implicit. However, they represent important linguistic intuition, they provide a uniform interface to semantics, and they are, as Schabes & Shieber (1994) argue, important in order to support statistical parameters in stochastic frameworks and appropriate adjunction constraints in TAG. In many frameworks, complementation and modification are in fact made explicit: LFG (Bresnan & Kaplan, 1982) provides a separate functional (f-) structure, and dependency grammars (see e.g. Mel'čuk (1988)) use these notions as the principal basis for syntactic representation. We will follow the dependency literature in referring to complementation and modification as syntactic dependency. As observed by Rambow and Joshi (1992), for TAG, the importance of the dependency structure means that not only the derived phrase-structure tree is of interest, but also the operations by which we obtained it from elementary structures. This information is encoded in the derivation tree (Vijay-Shanker, 1987).

However, as Vijay-Shanker (1992) observes, the TAG composition operations are not used uniformly: while substitution is used only to add a (nominal) complement, adjunction is used both for modification and (clausal) complementation. Clausal complementation could not be handled uniformly by substitution because of the existence of syntactic phenomena such as long-distance *wh*-movement in English. Furthermore, there is an inconsistency in

the directionality of the operations used for complementation in TAG@: nominal complements are substituted into their governing verb's tree, while the governing verb's tree is adjoined into its own clausal complement. The fact that adjunction and substitution are used in a linguistically heterogeneous manner means that (standard) TAG derivation trees do not provide a good representation of the dependencies between the words of the sentence, i.e., of the predicate-argument and modification structure.

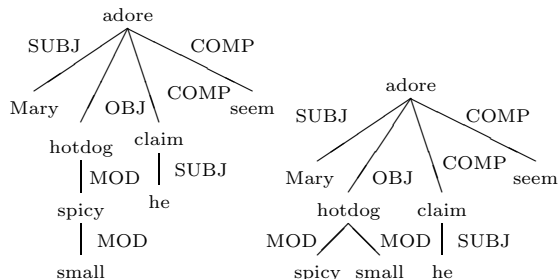


Figure 1: Derivation trees for (1): original definition (left); Schabes & Shieber definition (right)

For instance, English sentence (1) gets the derivation structure shown on the left in Figure 1¹.

(1) Small spicy hotdogs he claims Mary seems to adore

When comparing this derivation structure to the dependency structure in Figure 2, the following problems become apparent. First, both adjectives depend on *hotdog*, while in the derivation structure *small* is a daughter of *spicy*. In addition, *seem* depends on *claim* (as does its nominal argument, *he*), and *adore* depends on *seem*. In the derivation structure, *seem* is a daughter of *adore* (the direction does not express the actual dependency), and *claim* is also a daughter of *adore* (though neither is an argument of the other).

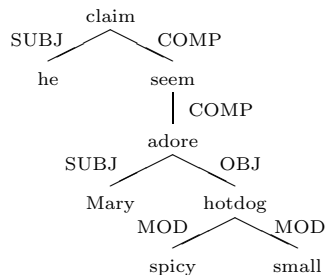


Figure 2: Dependency tree for (1)

Schabes & Shieber (1994) solve the first problem

¹For clarity, we depart from standard TAG notational practice and annotate nodes with lexemes and arcs with grammatical function.

by distinguishing between the adjunction of modifiers and of clausal complements. This gives us the derivation structure shown on the right in Figure 1. While this might provide a satisfactory treatment of modification at the derivation level, there are now three types of operations (two adjunctions and substitution) for two types of dependencies (arguments and modifiers), and the directionality problem for embedded clauses remains unsolved.

In defining DTG we have attempted to resolve these problems with the use of a single operation (that we call subserction) for handling all complementation and a second operation (called sister-adjunction) for modification. Before discussion these operations further we consider a second problem with TAG that has implications for the design of these new composition operations (in particular, subserction).

1.2 Problematic Constructions for TAG

TAG cannot be used to provide suitable analyses for certain syntactic phenomena, including long-distance scrambling in German (Becker et al., 1991), Romance Clitics (Bleam, 1994), *wh*-extraction out of complex picture-NPs (Kroch, 1987), and Kashmiri *wh*-extraction (presented here). The problem in describing these phenomena with TAG arises from the fact (observed by Vijay-Shanker (1992)) that adjoining is an overly restricted way of combining structures. We illustrate the problem by considering Kashmiri *wh*-extraction, drawing on Bhatt (1994). *Wh*-extraction in Kashmiri proceeds as in English, except that the *wh*-word ends up in sentence-second position, with a topic from the matrix clause in sentence-initial position. This is illustrated in (2a) for a simple clause and in (2b) for a complex clause.

- (2) a. rameshan kyaa dyutnay tse
Ramesh_{ERG} what_{NOM} gave you_{DAT}
What did you give Ramesh?
- b. rameshan kyaa_i chu baasaan [ki
Ramesh_{ERG} what is believe_{NPerf} that
me kor t_i]
I_{ERG} do
What does Ramesh believe that I did?

Since the moved element does not appear in sentence-initial position, the TAG analysis of English *wh*-extraction of Kroch (1987; 1989) (in which the matrix clause is adjoined into the embedded clause) cannot be transferred, and in fact no linguistically plausible TAG analysis appears to be available.

In the past, variants of TAG have been developed to extend the range of possible analyses. In Multi-Component TAG (MCTAG) (Joshi, 1987), trees are grouped into sets which must be adjoined together (multicomponent adjunction). However, MCTAG lack expressive power since, while syntactic relations are invariably subject to c-command or dominance constraints, there is no way to state that

two trees from a set must be in a dominance relation in the derived tree. MCTAG with Domination Links (MCTAG-DL) (Becker et al., 1991) are multi-component systems that allow for the expression of dominance constraints. However, MCTAG-DL share a further problem with MCTAG: the derivation structures cannot be given a linguistically meaningful interpretation. Thus, they fail to address the first problem we discussed (in Section 1.1).

1.3 The DTG Approach

Vijay-Shanker (1992) points out that use of adjunction for clausal complementation in TAG corresponds, at the level of dependency structure, to substitution at the foot node² of the adjoined tree. However, adjunction (rather than substitution) is used since, in general, the structure that is substituted may only form part of the clausal complement: the remaining substructure of the clausal complement appears above the root of the adjoined tree. Unfortunately, as seen in the examples given in Section 1.2, there are cases where satisfactory analyses cannot be obtained with adjunction. In particular, using adjunction in this way cannot handle cases in which parts of the clausal complement are required to be placed within the structure of the adjoined tree.

The DTG operation of subsertion is designed to overcome this limitation. Subsertion can be viewed as a generalization of adjunction in which components of the clausal complement (the subserted structure) which are not substituted can be interspersed within the structure that is the site of the subsertion. Following earlier work (Becker et al., 1991; Vijay-Shanker, 1992), DTG provide a mechanism involving the use of domination links (d-edges) that ensure that parts of the subserted structure that are not substituted dominate those parts that are. Furthermore, there is a need to constrain the way in which the non-substituted components can be interspersed³. This is done by either using appropriate feature constraints at nodes or by means of subsertion-insertion constraints (see Section 2).

We end this section by briefly commenting on the other DTG operation of sister-adjunction. In TAG, modification is performed with adjunction of modifier trees that have a highly constrained form. In particular, the foot nodes of these trees are always daughters of the root and either the leftmost or rightmost frontier nodes. The effect of adjoining a

tree of this form corresponds (almost) exactly to the addition of a new (leftmost or rightmost) subtree below the node that was the site of the adjunction. For this reason, we have equipped DTG with an operation (sister-adjunction) that does exactly this and nothing more. From the definition of DTG in Section 2 it can be seen that the essential aspects of Schabes & Shieber (1994) treatment for modification, including multiple modifications of a phrase, can be captured by using this operation⁴.

After defining DTG in Section 2, we discuss, in Section 3, DTG analyses for the English and Kashmiri data presented in this section. Section 4 briefly discusses DTG recognition algorithms.

2 Definition of D-Tree Grammars

A **d-tree** is a tree with two types of edges: domination edges (**d-edges**) and immediate domination edges (**i-edges**). D-edges and i-edges express domination and immediate domination relations between nodes. These relations are never rescinded when d-trees are composed. Thus, nodes separated by an i-edge will remain in a mother-daughter relationship throughout the derivation, whereas nodes separated by a d-edge can be equated or have a path of any length inserted between them during a derivation. D-edges and i-edges are not distributed arbitrarily in d-trees. For each internal node, either all of its daughters are linked by i-edges or it has a single daughter that is linked to it by a d-edge. Each node is labelled with a terminal symbol, a nonterminal symbol or the empty string. A d-tree containing n d-edges can be decomposed into $n + 1$ **components** containing only i-edges.

D-trees can be composed using two operations: **subsertion** and **sister-adjunction**. When a d-tree α is subserted into another d-tree β , a component of α is substituted at a frontier nonterminal node (a **substitution node**) of β and all components of α that are above the substituted component are inserted into d-edges above the substituted node or placed above the root node. For example, consider the d-trees α and β shown in Figure 3. Note that components are shown as triangles. In the composed d-tree γ the component $\alpha(5)$ is substituted at a substitution node in β . The components, $\alpha(1)$, $\alpha(2)$, and $\alpha(4)$ of α above $\alpha(5)$ drift up the path in β which runs from the substitution node. These components are then **inserted** into d-edges in β or above the root of β . In general, when a component $\alpha(i)$ of some d-tree α is inserted into a d-edge between nodes η_1 and η_2 two new d-edges are created, the first of which relates η_1 and the root node of $\alpha(i)$, and the second of which relates the frontier

²In these cases the foot node is an argument node of the lexical anchor.

³This was also observed by Rambow (1994a), where an integrity constraint (first defined for an ID/LP version of TAG (Becker et al., 1991)) is defined for a MCTAG-DL version called V-TAG. However, this was found to be insufficient for treating both long-distance scrambling and long-distance topicalization in German. V-TAG retains adjoining (to handle topicalization) for this reason.

⁴Santorini and Mahootian (1995) provide additional evidence against the standard TAG approach to modification from code switching data, which can be accounted for by using sister-adjunction.

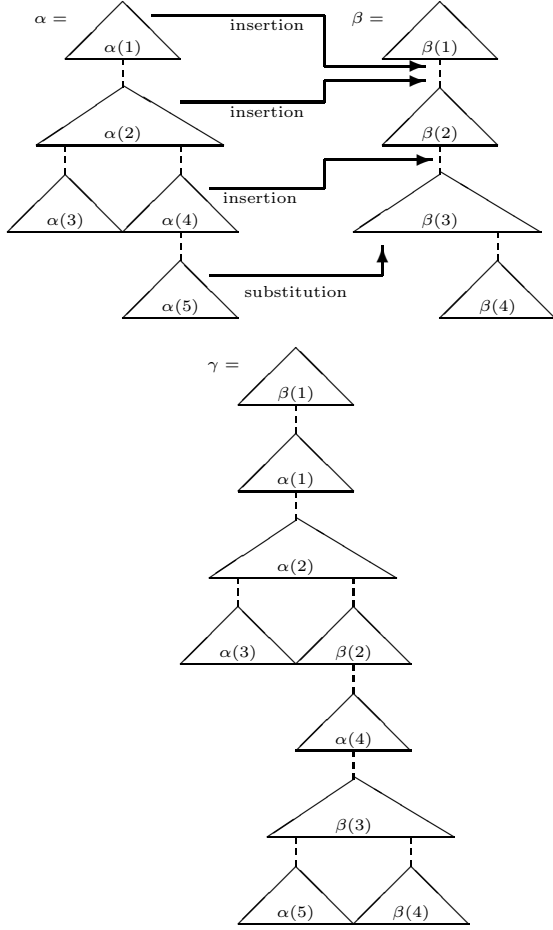


Figure 3: Subsertion

node of $\alpha(i)$ that dominates the substituted component to η_2 . It is possible for components above the substituted node to drift arbitrarily far up the d-tree and distribute themselves within domination edges, or above the root, in any way that is compatible with the domination relationships present in the substituted d-tree. DTG provide a mechanism called **subsertion-insertion constraints** to control what can appear within d-edges (see below).

The second composition operation involving d-trees is called sister-adjunction. When a d-tree α is sister-adjoined at a node η in a d-tree β the composed d-tree γ results from the addition to β of α as a new leftmost or rightmost sub-d-tree below η . Note that sister-adjunction involves the addition of exactly one new immediate domination edge and that several sister-adjunctions can occur at the same node. **Sister-adjoining constraints** specify where d-trees can be sister-adjoined and whether they will be right- or left-sister-adjoined (see below).

A DTG is a four tuple $G = (V_N, V_T, S, D)$ where V_N and V_T are the usual nonterminal and termi-

nal alphabets, $S \in V_N$ is a distinguished nonterminal and D is a finite set of **elementary** d-trees. A DTG is said to be **lexicalized** if each d-tree in the grammar has at least one terminal node. The elementary d-trees of a grammar G have two additional annotations: subsertion-insertion constraints and sister-adjoining constraints. These will be described below, but first we define simultaneously DTG derivations and subsertion-adjoining trees (SA-trees), which are partial derivation structures that can be interpreted as representing dependency information, the importance of which was stressed in the introduction⁵.

Consider a DTG $G = (V_N, V_T, S, D)$. In defining SA-trees, we assume some naming convention for the elementary d-trees in D and some consistent ordering on the components and nodes of elementary d-trees in D . For each i , we define the set of d-trees $T_i(G)$ whose derivations are captured by SA-trees of height i or less. Let $T_0(G)$ be the set D of elementary d-trees of G . Mark all of the components of each d-tree in $T_0(G)$ as being **substitutable**⁶. Only components marked as substitutable can be substituted in a subsertion operation. The SA-tree for $\alpha \in T_0(G)$ consists of a single node labelled by the elementary d-tree name for α .

For $i > 0$ let $T_i(G)$ be the union of the set $T_{i-1}(G)$ with the set of all d-trees γ that can be produced as follows. Let $\alpha \in D$ and let γ be the result of subserting or sister-adjoining the d-trees $\gamma_1, \dots, \gamma_k$ into α where $\gamma_1, \dots, \gamma_k$ are all in $T_{i-1}(G)$, with the subsertions taking place at different substitution nodes in α as the footnote. Only substitutable components of $\gamma_1, \dots, \gamma_k$ can be substituted in these subsertions. Only the new components of γ that came from α are marked as substitutable in γ . Let τ_1, \dots, τ_k be the SA-trees for $\gamma_1, \dots, \gamma_k$, respectively. The SA-tree τ for γ has root labelled by the name for α and k subtrees τ_1, \dots, τ_k . The edge from the root of τ to the root of the subtree τ_i is labelled by l_i ($1 \leq i \leq k$) defined as follows. Suppose that γ_i was subserted into α and the root of τ_i is labelled by the name of some $\alpha' \in D$. Only components of α' will have been marked as substitutable in γ_i . Thus, in this subsertion some component $\alpha'(j)$ will have been substituted at a node in α with address n . In this case, the label l_i is the pair (j, n) . Alternatively, γ_i will have

⁵Due to space limitations, in the following definitions we are forced to be somewhat imprecise when we identify a node in a derived d-tree with the node in the elementary d-trees (elementary nodes) from which it was derived. This is often done in TAG literature, and hopefully it will be clear what is intended.

⁶We will discuss the notion of substitutability further in the next section. It is used to ensure the SA-tree is a tree. That is, an elementary structure cannot be subserted into more than one structure since this would be counter to our motivations for using subsertion for complementation.

been d -sister-adjoined at some node with address n in α , in which case l_i will be the pair (d, n) where $d \in \{\text{left}, \text{right}\}$.

The **tree set** $T(G)$ generated by G is defined as the set of trees γ such that: $\gamma' \in T_i(G)$ for some $i \geq 0$; γ' is rooted with the nonterminal S ; the frontier of γ' is a string in V_T^* ; and γ results from the removal of all d -edges from γ' . A d -edge is removed by merging the nodes at either end of the edge as long as they are labelled by the same symbol. The **string language** $L(G)$ associated with G is the set of terminal strings appearing on the frontier of trees in $T(G)$.

We have given a reasonably precise definition of SA-trees since they play such an important role in the motivation for this work. We now describe informally a structure that can be used to encode a DTG derivation. A derivation graph for $\gamma \in T(G)$ results from the addition of insertion edges to a SA-tree τ for γ . The location in γ of an inserted elementary component $\alpha(i)$ can be unambiguously determined by identifying the source of the node (say the node with address n in the elementary d-tree α') with which the root of this occurrence of $\alpha(i)$ is merged with when d -edges are removed. The insertion edge will relate the two (not necessarily distinct) nodes corresponding to appropriate occurrences of α and α' and will be labelled by the pair (i, n) .

Each d -edge in elementary d-trees has an associated subserction-insertion constraint (SIC). A SIC is a finite set of elementary node addresses (ENAs). An ENA η specifies some elementary d-tree $\alpha \in D$, a component of α and the address of a node within that component of α . If a ENA η is in the SIC associated with a d -edge between η_1 and η_2 in an elementary d-tree α then η cannot appear properly within the path that appears from η_1 to η_2 in the derived tree $\gamma \in T(G)$.

Each node of elementary d-trees has an associated sister-adjunction constraint (SAC). A SAC is a finite set of pairs, each pair identifying a direction (left or right) and an elementary d-tree. A SAC gives a complete specification of what can be sister-adjoined at a node. If a node η is associated with a SAC containing a pair (d, α) then the d-tree α can be d -sister-adjoined at η . By definition of sister-adjunction, all substitution nodes and all nodes at the top of d -edges can be assumed to have SACs that are the empty-set. This prevents sister-adjunction at these nodes.

In this section we have defined “raw” DTG. In a more refined version of the formalism we would associate (a single) finite-valued feature structure with each node⁷. It is a matter of further research to determine to what extent SICs and SACs can be stated globally for a grammar, rather than being attached

to d -edges/nodes⁸. See the next section for a brief discussion of linguistic principles from which a grammar’s SICs could be derived.

3 Linguistic Examples

In this section, we show how an account for the data introduced in Section 1 can be given with DTG.

3.1 Getting Dependencies Right: English

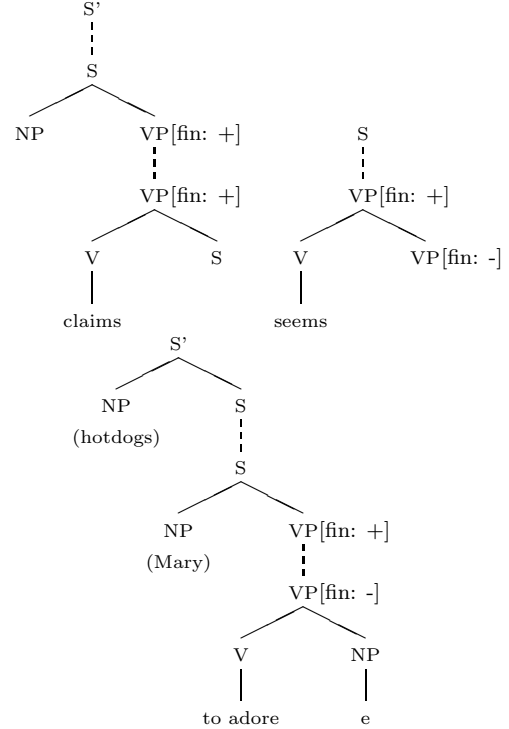


Figure 4: D-trees for (1)

In Figure 4, we give a DTG that generates sentence (1). Every d-tree is a projection from a lexical anchor. The label of the maximal projection is, we assume, determined by the morphology of the anchor. For example, if the anchor is a finite verb, it will project to S , indicating that an overt syntactic (“surface”) subject is required for agreement with it (and perhaps case-assignment). Furthermore, a finite verb may optionally also project to S' (as in the d-tree shown for *claims*), indicating that a *wh*-moved or topicalized element is required. The finite verb *seems* also projects to S , even though it does not itself provide a functional subject. In the case of the *to adore* tree, the situation is the inverse: the functional subject requires a finite verb

⁷Trees used in Section 3 make use of such feature structures.

⁸In this context, it might be beneficial to consider the expression of a feature-based lexicalist theory such as HPSG in DTG, similar to the compilation of HPSG to TAG (Kasper et al., 1995).

to agree with, which is signaled by the fact that its component’s root and frontier nodes are labelled S and VP, respectively, but the verb itself is not finite and therefore only projects to VP[-fin]. Therefore, the subject will have to raise out of its clause for agreement and case assignment. The direct object of *to adore* has *wh*-moved out of the projection of the verb (we include a trace for the sake of clarity).

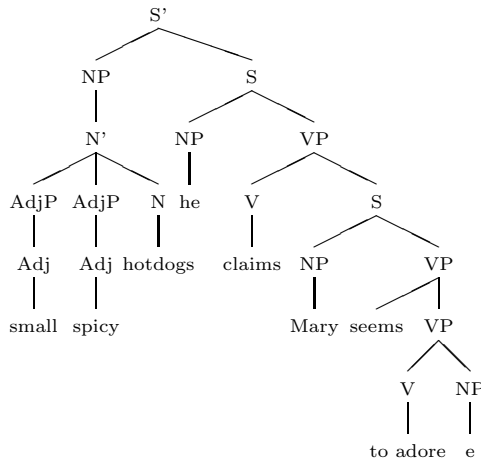


Figure 5: Derived tree for (1)

We add SICs to ensure that the projections are respected by components of other d-trees that may be inserted during a derivation. A SIC is associated with the d-edge between VP and S node in the *seems* d-tree to ensure that no node labelled S' can be inserted within it – i.e., it can not be filled by with a *wh*-moved element. In contrast, since both the subject and the object of *to adore* have been moved out of the projection of the verb, the path to these arguments do not carry any SIC at all⁹.

We now discuss a possible derivation. We start out with the most deeply embedded clause, the *adores* clause. Before subserting its nominal arguments, we sister-adjoin the two adjectival trees to the tree for *hotdogs*. This is handled by a SAC associated with the N' node that allows all trees rooted in AdjP to be left sister-adjoined. We then subsert this structure and the subject into the *to adore* d-tree. We subsert the resulting structure into the *seems* clause by substituting its maximal projection node, labelled VP[fin: -], at the VP[fin: -] frontier node of *seems*, and by inserting the subject into the d-edge of the *seems* tree. Now, only the S node of the *seems* tree (which is its maximal projection) is substitutable. Finally, we subsert this derived struc-

⁹We enforce island effects for *wh*-movement by using a [\pm extract] feature on substitution nodes. This corresponds roughly to the analysis in TAG, where islandhood is (to a large extent) enforced by designating a particular node as the foot node (Kroch & Joshi, 1986).

ture into the *claims* d-tree by substituting the S node of *seems* at the S complement node of *claims*, and by inserting the object of *adores* (which has not yet been used in the derivation) in the d-edge of the *claims* d-tree above its S node. The derived tree is shown in Figure 5. The SA-tree for this derivation corresponds to the dependency tree given previously in Figure 2.

Note that this is the only possible derivation involving these three d-trees, modulo order of operations. To see this, consider the following putative alternate derivation. We first subsert the *to adore* d-tree into the *seems* tree as above, by substituting the anchor component at the substitution node of *seems*. We insert the subject component of *to adore* above the anchor component of *seems*. We then subsert this derived structure into the *claims* tree by substituting the root of the subject component of *to adore* at the S node of *claims* and by inserting the S node of the *seems* d-tree as well as the object component of the *to adore* d-tree in the S'/S d-edge of the *claims* d-tree. This last operation is shown in Figure 6. The resulting phrase structure tree would be the same as in the previously discussed derivation, but the derivation structure is linguistically meaningless, since *to adore* would have been subserted into both *seems* and *claims*. However, this derivation is ruled out by the restriction that only substitutable components can be substituted: the subject component of the *adore* d-tree is not substitutable after subsertion into the *seems* d-tree, and therefore it cannot be substituted into the *claims* d-tree.

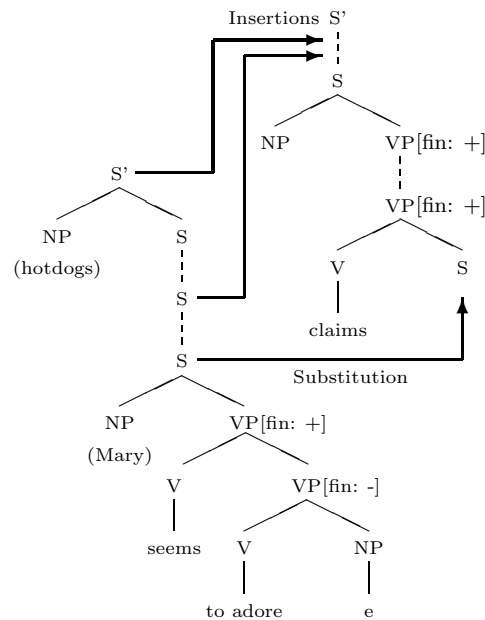


Figure 6: An ill-formed derivation

In the above discussion, substitutability played a

central role in ruling out the derivation. We observe in passing that the SIC associated to the d-edge in the *seems* d-tree also rules out this derivation. The derivation requires that the S node of *seems* be inserted into the S'/S d-edge of *claims*. However, we would have to stretch the edge over two components which are both ruled out by the SIC, since they violate the projection from *seems* to its S node. Thus, the derivation is excluded by the independently motivated SICs, which enforce the notion of projection. This raises the possibility that, in grammars that express certain linguistic principles, substitutability is not needed for ruling out derivations of this nature. We intend to examine this issue in future work.

3.2 Getting Word Order Right: Kashmiri

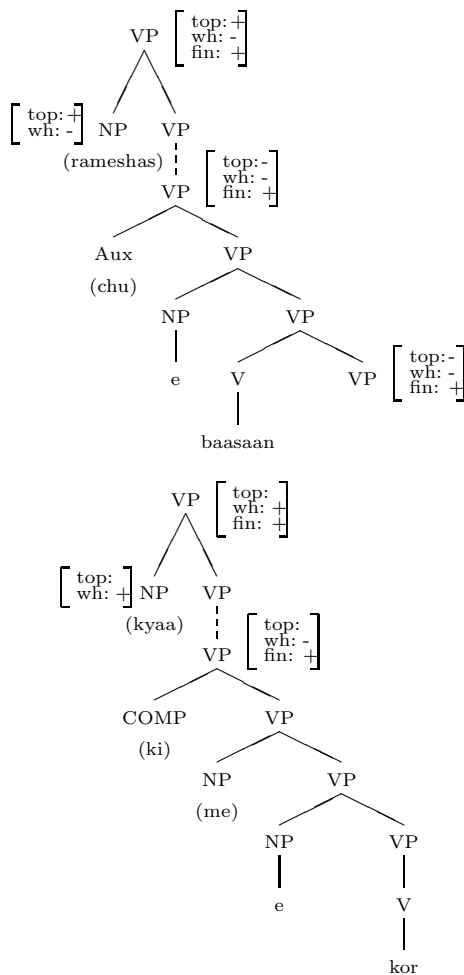


Figure 7: D-trees for (2b)

Figure 7 shows the matrix and embedded clauses for sentence (2b). We use the node label VP throughout and use features such as *top* (for topic) to differentiate different levels of projection. Observe that in both trees an argument has been fronted.

Again, we will use the SICs to enforce the projection from a lexical anchor to its maximal projection. Since the direct object of *kor* has *wh*-moved out of its clause, the d-edge connecting it to the maximal projection of its verb has no SIC. The d-edge connecting the maximal projection of *baasaan* to the Aux component, however, has a SIC that allows only VP[*wh*: +, *top*: -] nodes to be inserted.

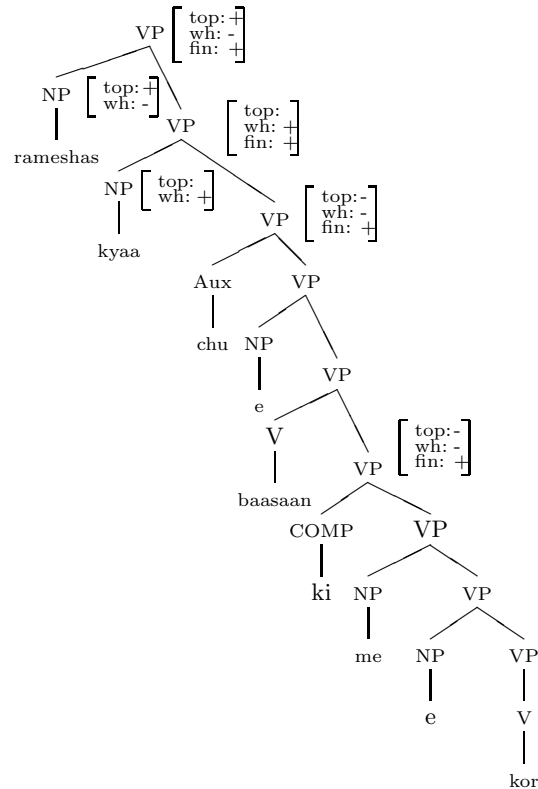


Figure 8: Derived d-tree for (2b)

The derivation proceeds as follows. We first subvert the embedded clause tree into the matrix clause tree. After that, we subvert the nominal arguments and function words. The derived structure is shown in Figure 8. The associated SA-tree is the desired, semantically motivated, dependency structure: the embedded clause depends on the matrix clause.

In this section, we have discussed examples where the elementary objects have been obtained by projecting from lexical items. In these cases, we overcome both the problems with TAG considered in Section 1. The SICs considered here enforce the same notion of projection that was used in obtaining the elementary structures. This method of arriving at SICs not only generalizes for the English and Kashmiri examples but also appears to apply to the case of long-distance scrambling and topicalization in German.

4 Recognition

It is straightforward to adapt the polynomial-time CKY-style recognition algorithm for a lexicalized UVG-DL of Rambow (1994b) for DTG. The entries in this array recording derivations of substrings of input contain a set of elementary nodes along with a multi-set of components that must be inserted above during bottom-up recognition. These components are added or removed at substitution and insertion. The algorithm simulates traversal of a derived tree; checking for SICs and SACs can be done easily. Because of lexicalization, the size of these multi-sets is polynomially bounded, from which the polynomial time and space complexity of the algorithm follows.

For practical purposes, especially for lexicalized grammars, it is preferable to incorporate some element of prediction. We are developing a polynomial-time Earley style parsing algorithm. The parser returns a parse forest encoding all parses for an input string. The performance of this parser is sensitive to the grammar and input. Indeed it appears that for grammars that lexicalize CFG and for English grammar (where the structures are similar to the LTAG developed at University of Pennsylvania (XTAG Research Group, 1995)) we obtain cubic-time complexity.

5 Conclusion

DTG, like other formalisms in the TAG family, is lexicalizable, but in addition, its derivations are themselves linguistically meaningful. In future work we intend to examine additional linguistic data, refining aspects of our definition as needed. We will also study the formal properties of DTG, and complete the design of the Earley style parser.

Acknowledgements

We would like to thank Rakesh Bhatt for help with the Kashmiri data. We are also grateful to Tilman Becker, Gerald Gazdar, Aravind Joshi, Bob Kasper, Bill Keller, Tony Kroch, Klaus Netter and the ACL-95 referees. Rambow was supported by the North Atlantic Treaty Organization under a Grant awarded in 1993, while at TALANA, Université Paris 7.

References

- [Becker et al.1991] T. Becker, A. Joshi, & O. Rambow. 1991. Long distance scrambling and tree adjoining grammars. In *EACL-91*, 21–26.
- [Bhatt1994] R. Bhatt. 1994. *Word order and case in Kashmiri*. Ph.D. thesis, Univ. Illinois.
- [Bleam1994] T. Bleam. 1994. Clitic climbing in spanish: a GB perspective. In TAG+ Workshop, Tech. Rep. TALANA-RT-94-01, Université Paris 7, 16–19.
- [Bresnan & Kaplan1982] J. Bresnan & R. Kaplan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan, ed., *The Mental Representation of Grammatical Relations*. MIT Press.
- [Frank1992] R. Frank. 1992. *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives*. Ph.D. thesis, Dept. Comp. & Inf. Sc., Univ. Pennsylvania.
- [Joshi1987] A. Joshi. 1987. An introduction to tree adjoining grammars. In A. Manaster-Ramer, ed., *Mathematics of Language*, 87–114.
- [Joshi et al.1975] A. Joshi, L. Levy, & M. Takahashi. 1975. Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10(1):136–163.
- [Joshi & Schabes1991] A. Joshi & Y. Schabes. 1991. Tree-adjoining grammars and lexicalized grammars. In M. Nivat & A. Podelski, eds., *Definability and Recognizability of Sets of Trees*.
- [Kasper et al.1995] R. Kasper, B. Kiefer, K. Netter, & K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. In ACL-95.
- [Kroch1987] A. Kroch. 1987. Subjacency in a tree adjoining grammar. In A. Manaster-Ramer, ed., *Mathematics of Language*, 143–172.
- [Kroch1989] A. Kroch. 1989. Asymmetries in long distance extraction in a Tree Adjoining Grammar. In Mark Baltin & Anthony Kroch, editors, *Alternative Conceptions of Phrase Structure*, 66–98.
- [Kroch & Joshi1986] A. Kroch & A. Joshi. 1986. Analyzing extraposition in a tree adjoining grammar. In G. Huck & A. Ojeda, eds., *Syntax & Semantics: Discontinuous Constituents*, 107–149.
- [Mel'čuk1988] I. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*.
- [Rambow1994a] O. Rambow. 1994. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, Dept. Comput. & Inf. Sc., Univ. Pennsylvania.
- [Rambow1994b] O. Rambow. 1994. Multiset-Valued Linear Index Grammars. In ACL-94, 263–270.
- [Rambow & Joshi1992] O. Rambow & A. Joshi. 1992. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In *Intern. Workshop on The Meaning-Text Theory*, Darmstadt. Arbeitspapiere der GMD 671, 47–66.
- [Santorini and Mahootian1995] B. Santorini & S. Mahootian. 1995. Codeswitching and the syntactic status of adnominal adjectives. *Lingua*, 95.
- [Schabes & Shieber1994] Y. Schabes & S. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Comput. Ling.*, 20(1):91–124.
- [Shieber1985] S. Shieber. 1985. Evidence against the context-freeness of natural language. *Ling. & Phil.*, 8:333–343.
- [Vijay-Shanker1987] K. Vijay-Shanker. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, Dept. Comput. & Inf. Sc., Univ. Pennsylvania.

- [Vijay-Shanker1992] K. Vijay-Shanker. 1992. Using descriptions of trees in a tree adjoining grammar. *Comput. Ling.*, 18(4):481–517.
- [XTAG Research Group1995] The XTAG Research Group. 1995. A lexicalized tree adjoining grammar for English. Tech. Rep. IRCS Report 95-03, Univ. Pennsylvania.